Security: Authorization in Mac OS X

Session 110

Security: Authorization in Mac OS X

Craig Keithley
Security and Cryptography Technology Evangelist

Introduction

- Discuss the recommended usage of the Authorization Services
- Make Mac OS X more secure and configurable by adopting this API

Security: Authorization in Mac OS X

Michael Brouwer
Technical Lead Data Security Group

What You Will Learn

- Learn to build applications that need privileged system access
- Learn to build installers that can install privileged application components
- Learn ways to access privileged system calls without running your entire application with root privileges

What This Session Will Cover

- Understanding Authorization
- What it does and does not do
- Authorization Services
- Examples of how to not use Authorization Services
- Examples of how to use Authorization Services

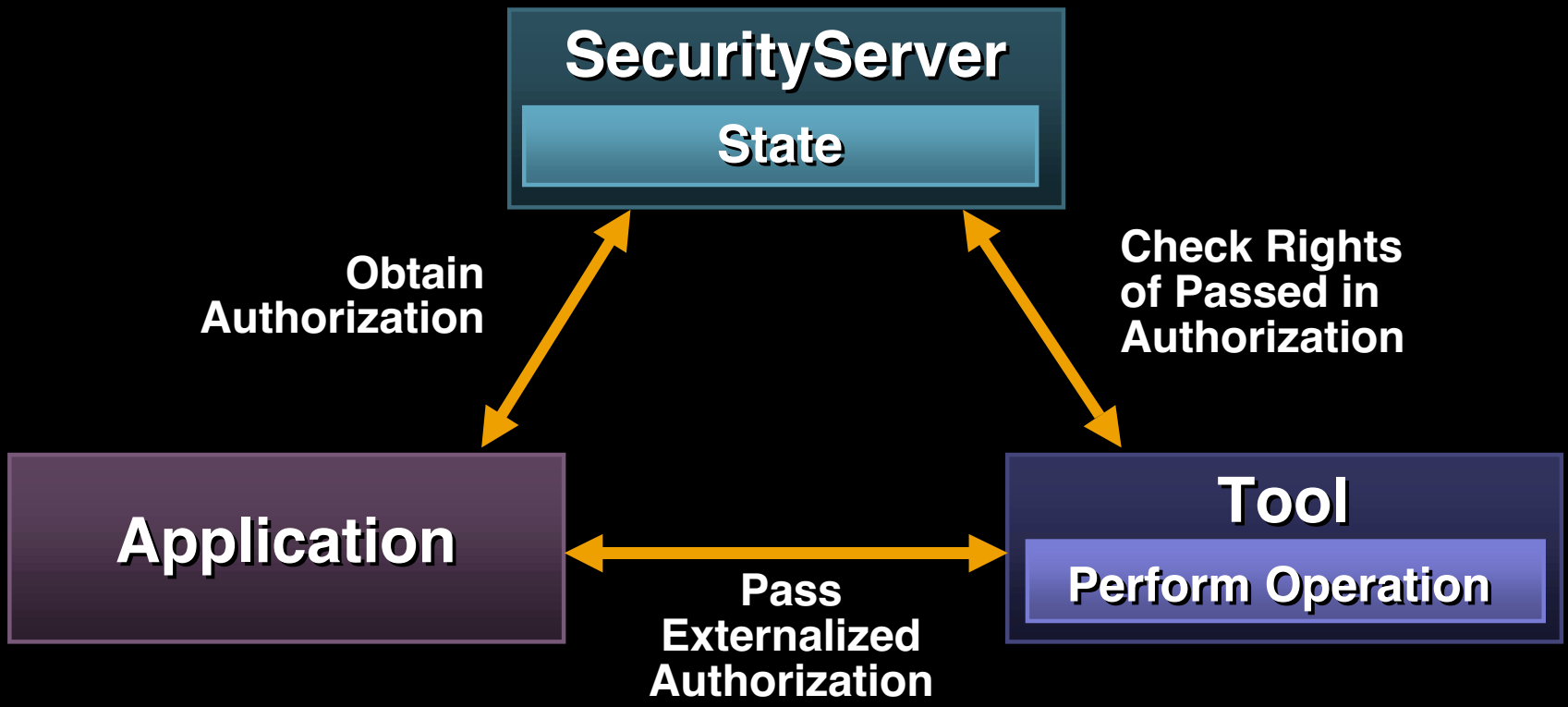
Understanding Authorization

- Understanding Authorization vs. Authentication
- Why use Authorization Services
    - Better than conventional ad-hoc methods for performing Authorization
    - Make an explicit decision and allow for fine grain control of who can do what

Benefits of Centralized Authorization

- Easier to audit
- Configurable/securable
- Flexible
- Explicit vs. ad hoc

Recommended Architecture



What It Does

- Password based only
- Using Directory Services as of Jaguar
- Answers the “Do I have the right to do this” question
- Provides user interaction when necessary

What It Does Not Do

- Answers the “Do I have the right to do this” question, but does not grant access
- Mac OS X is not a capability-based operating system

Case Study

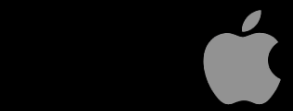
- Emptying the trash when it contains directories that are not writable by the current user

Authorization Services

- Naming of rights
- Application communication
- Recommended usage of Authorization Services
- Temporary solution for third-party installers

Naming of Rights

- Rights are in a hierarchical namespace
- We define domain, but not individual rights since each application has different needs
- Example right names:
  - **system.login.console**
  - **system.login.pam**
  - **system.device.dvd.setregion.change**
  - **system.device.dvd.setregion.initial**
  - **system.preferences**
  - **system.privilege.admin**



Application Communication

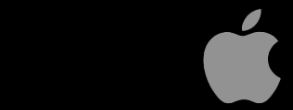
- Split off security sensitive operations into small well understood tools
- Passing Authorization tokens between processes

Recommended Usage of Authorization API (App)

- Create an AuthorizationRef
- Optional
  - Use CopyRights to figure out what is currently allowed
  - When the user wishes to make changes, perform a pre-authorization using CopyRights and ask for the rights you need
- Pass the AuthorizationRef to a privileged tool/daemon by Externalizing the AuthorizationRef

Recommended Usage of Authorization API (Tool)

- Call CreateFromExternal to obtain the AuthorizationRef passed in by the application
- Use CopyRights to determine whether the right needed has been granted
- Perform the requested operation(s) (if permitted by CopyRights)



Preauthorization

- Application may preauthorize for the rights needed
- The tool performing the operation should call CopyRights again without the preauthorize flag
- Keep the final call to CopyRights as close as possible to the operation
- Allows addition of audit logging
- Allows for zero length timeouts or removable tokens

Temporary Solution for Third-Party Installers

- AuthorizationExecuteWithPrivileges
- Right required:
  - **system.privilege.admin**

Examples of How to Not Use Authorization Services

- Do not run commands from BSD or shell scripts and *never* call **system()** or **popen()**
- Do not run your entire Application as root
- Do not rely on credentials being shared or timeouts being greater than 0

Examples of How to Use Authorization Services

- When using **AuthorizationExecWithPrivileges()** call it at most once
- Test with the most secure possible /etc/authorization
- Define rights for user initiated operations, not for low-level system operations
  - **system.finder.empty.trash (good)**
  - **system.finder.delete.file (bad)**

Most Secure Version of /etc/authorization Possible

```
<!DOCTYPE plist PUBLIC "-//Apple Computer//DTD PLIST 1.0//EN”
    "http://www.apple.com/DTDs/PropertyList-1.0.dtd">

<plist version="1.0">
<dict><key/><dict>
    <key>group</key><string>admin</string>
    <key>shared</key><false/>
    <key>timeout</key><integer>0</integer>
</dict></dict></plist>
```

Demo

Summary

- "Rights" are not capabilities
- Do not run an entire Application with root privileges, limit this to a small tool
- Create an **AuthorizationRef** and pass it to the tool
- Keep the privileged tool as small as possible and audit it
- Do not run commands from BSD or shell scripts and *never* **call system()** or **popen()**

Resources

Security
Specifications and SDKs
http://developer.apple.com/macos/security.html

CDSA 2.0
Specifications
http://www.opengroup.org

PC/SC
Specifications
http://www.pcscworkgroup.com

Roadmap

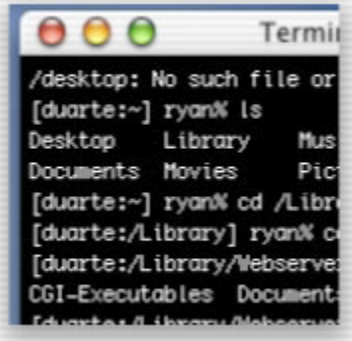
113 Security: CDSA and Secure Transport Common Data Security Architecture	Civic **Thurs., 9:00am**
114 Security: Certificates in Mac OS X Using X.509 certificates on Mac OS X	Civic **Thurs., 10:30am**
814 Kerberos in Mac OS X Learn about Kerberos on Mac OS X	Room C **Thurs., 5:00pm**
FF006 Security Give us your feedback on security issues	Room J1 **Thurs., 2:00pm**

Who to Contact

Craig Keithley
Security and Cryptography Technology Evangelist
keithley@apple.com

http://developer.apple.com/wwdc2002/urls.html

Q&A



Craig Keithley
Security and Cryptography Technology Evangelist
keithley@apple.com

http://developer.apple.com/wwdc2002/urls.html


WWDC2002


WWDC2002


WWDC2002